Laboratorium 3. Stemming, lematyzacja, worek słów, taka klasyfikacja

Poprzednie zajęcia pozwoliły poznać narzędzia do gromadzenia danych. Mając dane oznaczone etykietami, można wykonać prosty eksperyment klasyfikacji. Jednakże, surowy tekst napisany w języku naturalnym nie nadaje się do trenowania modeli. Aby było to możliwe konieczna jest wektoryzacja danych. Rozwiązanie tego laboratorium pozwoli poznać przykładowy sposób transformacji surowych danych tekstowych z etykietami do prostego modelu zdolnego do klasyfikacji.

Zadanie 3.1. Przygotowanie danych

Wczytaj dane z katalogów `positive` oraz `negative` oraz dla każdego dokumentu:

1. Zamień w tekście wszystkie duże litery na małe
2. Dokonaj tokenizacji z użyciem `TreebankWordTokenizer`
3. Usuń z tokenów `stopwords`[1] (możesz też usunąć inne tokeny, które nie mają znaczenia np. "
")
4. Wykonaj stemming na tokenach z poprzedniego punktu z użyciem `PorterStemmer`
5. Wykonaj lematyzację na tokenach z punktu 3. z użyciem `WordNetLemmatizer`
6. Zapisz do tablic cztery zestawy tokenów:
   - Tokeny
   - Bez stop words
   - Po stemmingu
   - Po lematyzacji

Dokumentacja

- `TreebankWordTokenizer`[2]
- `PorterStemmer`[3]
- `WordNetLemmatizer`[4]

Zadanie 3.2. Worek słów

Wykonaj wektoryzację dla każdego z czterech zestawów tokenów z zadania 3.1, zgodnie z podejściem "Bag of Words"

- Początkowo zapisz dane do zwykłej listy
- Wykorzystaj funkcję `Counter` w celu wyznaczenia występowania słów rozszerzając listę
- Zamień listę na macierz DataFrame wykorzystując funkcję `from_records`
- Zamień wartości "Nan" na zera oraz przekonwertuj na macierz typu `ndarray`[5]

Dokumentacja

- `Counter`[6]
- `from_records`[7]
- `fillna`[8]

Zadanie 3.3. Klasyfikacja

1. Utwórz tablicę etykiet (300 zer oraz 300 jedynek) typu `ndarray`
2. Podziel dane na treningowe (70%) oraz testowe (30%) z użyciem `train_test_split`)
3. Wytrenuj model z użyciem klasyfikatora `MultinomialNB` na danych treningowych (funkcja `fit()`)
4. Dokonaj predykcji modelu na danych testowych i zapisz wynik (funkcja `predict()`)
5. Wyświetl dokładności wyrażoną za pomocą metryki `accuracy_score`

Dokumentacja

- `train_test_split`[9]
- `MultinomialNB`[10]
- `accuracy_score`[11]

[1] <https://www.nltk.org/book/ch02.html>
[2] <https://www.nltk.org/api/nltk.tokenize.treebank.html>
[3] <https://www.nltk.org/api/nltk.stem.porter.html#module-nltk.stem.porter>
[4] <https://www.nltk.org/api/nltk.stem.wordnet.html>
[5] <https://numpy.org/doc/stable/reference/generated/numpy.array.html>
[6] <https://docs.python.org/3/library/collections.html#collections.Counter>
[7] <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.from_records.html?highlight=from_records>
[8] <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html>
[9] <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>
[10] <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html>
[11] <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html>